САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ И МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Сень Анастасия Игоревна

Выпускная квалификационная работа бакалавра

Разработка инструментария предварительной обработки данных для скрининга технологий МУН

Направление 02.03.02

Фундаментальная информатика и информационные технологии

Научный руководитель,
доктор технических наук,
профессор
Дегтярев А. Б.

Санкт-Петербург

2018

Содержание

Введение

Одной из важнейших тенденций современной мировой нефтедобывающей промышленности является снижение производства на легкодоступных и высокопроизводительных нефтяных месторождениях. Чтобы сохранить и увеличить текущие объемы производства, необходимо проводить более подробный анализ проблемы глубокого бурения. В нашей стране тяжелая нефть имеет особое значение.

За последние пятнадцать лет истощение запасов легкой нефти привело к снижению темпов и осложнениям условий добычи, а, следовательно, к сокращению объемов добычи нефти и уменьшению периода капитального ремонта и срока эксплуатации оборудования. Частая смена и использование малопродуктивного оборудования сопровождается ростом себестоимости. Следствием этих процессов является устойчивое снижение рентабельности добычи нефти [1].

В России легкие нефтяные месторождения разрабатываются более чем наполовину, тогда как геологические запасы высоковязкой и тяжелой нефти достигают 6-7 млрд. тонн (40-50 млрд. Баррелей). Однако использование и производство этих запасов требуют специализированных дорогостоящих технологий [2]. Эксплуатационные расходы на добычу тяжелой нефти и природного битума превышают темпы добычи легкой нефти в 3-4 раза. Это связано не только с высокой плотностью и вязкостью тяжелых масел, но и с недостаточным развитием технологий для их производства и переработки. Таким образом, исследование и применение новых технологий производства тяжелой и высоковязкой нефти является приоритетным направлением развития нефтедобывающего сектора.

Методы увеличения нефтеотдачи (МУН) выбираются в зависимости от геолого-физических характеристик и свойств месторождения. Не существует

полностью идентичных месторождений, они могут различаться по плотности, вязкости и составу нефти; по нефтенасыщенности, литологическому составу, смачиваемости, глубине и толщине, горизонтальной проницаемости коллектора и т.д. При этом на части залежи технология может быть применима, а в целом по пласту – нет. Следовательно, для каждой скважины необходимо оптимальным образом спрогнозировать соответствующую технологию.

Скрининг обеспечивает поиск наиболее подходящих технологий путем анализа существующих МУН для заданных характеристик конкретной залежи. В настоящее время большинство результатов исследований храниться в различных электронных базах в виде научных публикаций. Таким образом, скрининг технологий МУН сводится к изучению статей, посвященных повышению нефтедобычи.

В 2007 году американским ученым Джимом Греем было введено понятие четвертой парадигмы научных исследований, которая встает в один ряд с двумя классическими – теорией и экспериментом, а также третьей – крупномасштабным компьютерным моделированием. Четвертая парадигма подразумевает совместное использование моделирования, теории и эксперимента в условиях огромных объемов данных [3], с последующей обработкой и хранением полученной информации (так называемый *архив науки*).

Архив науки возможно использовать в *большом масштабе*, как корпус текстов и набор взаимосвязанных источников данных. При применении вычислительных технологий, производится поиск удовлетворяющих источников и выдвижение гипотез на основе комбинаций найденных данных об экспериментах, теориях и моделировании. Таким образом, согласно парадигме Грея, можно сказать, что все уже заранее известно и посчитано, нужно лишь только найти. Для этого существуют различные распределенные

базы данных, электронные библиотеки, облачные хранилища и т.п. Они содержат огромный цифровой ресурс, который стал важной архитектурой веб-приложений.

Данные цифровых библиотек и баз данных огромны, неоднородны, динамичны. Точка доступа к данным — это способ представления запроса гетерогенной информации и предоставления интеллектуальных услуг пользователям. Электронные базы направлены на создание широкомасштабной расширяемой и распределенной системы знаний путем интеграции и использования новейших компьютерных, коммуникационных и мультимедийных технологий. Благодаря системе знаний электронная база может обеспечить высокоскоростной унифицированный поиск услуги электронного доступа для своих пользователей.

Основой данной работы является исследование методов извлечения информации из электронных ресурсов, а также проведение анализа полученной информации для выявления наиболее релевантных источников. Работа выполнена по заказу ООО «Газпромнефть НТЦ»; в качестве метода анализа источников компанией предложен поиск центров компетенций. Под данным поиском понимается выделение компаний, наиболее грамотных и экспертных в определённой области знаний.

Постановка задачи

Целью данной работы является повышение эффективности скрининга технологий МУН. Для достижения обозначенной цели необходимо выполнить следующие задачи:

1. обзор и модификация современных методов извлечения информации из электронных ресурсов;
2. разработка программы извлечения данных;
3. проведение анализа полученных данных.

Главная задача данного программного обеспечения – автоматизировать извлечение и анализ необходимой для скрининга информации.

Обзор литературы

Для подготовки к написанию данной работы было изучено немало материалов. Основой извлечения данных из электронных интернет-ресурсов является анализ исходного кода страниц. В статье *«A Survey on Region Extractors from Web Documents», Hassan A. Sleiman* [4] описываются общие подходы к извлечению информации из веб-документов. В работе использовалась часть введения статьи для определения типов извлечения. Экстракторы информации могут быть в целом классифицированы в зависимости от того, имеют ли они дело с текстовыми или полуструктурированными веб-документами. В случае с текстом требуются методы обработки естественного языка, тогда как для полуструктурированных веб-документов используют теги HTML.

Поскольку исходный код электронных библиотек является полуструктурированным, далее будут рассмотрены алгоритмы извлечения данных именно для таких документов.

В *«Record-Boundary Discovery in Web Documents», Embley et al.* [5] предложен алгоритм для извлечения записей из самой большой области данных в веб-документе. Он основывается на гипотезе о том, что существует уникальная область данных, которая является самой большой в веб-документе, и она содержит наибольшее количество непосредственно дочерних узлов. Для выбора границ записей оценивается вероятность стандартного отклонения размера информации.

В *«A fully automated object extraction system for the World Wide Web», Buttler et al.* [6] рассмотрен алгоритм OMINI, в котором для определения тегов-разделителей, вместо стандартного отклонения, используется комбинация пяти эвристик: стандартного отклонения, повторяющегося шаблона, идентифицируемого разделителя (сверка по перечню часто используемых),

тегов-сестёр (подсчет вхождения пар тегов) и частичного пути (ранжирование по числу идентифицированных путей в узле).

В статье *«Data-rich section extraction from HTML pages», Wang* [7] описывает алгоритм Data-rich section extraction (DSE) суть которого заключается в том, что страницы с одного и того же веб-сайта часто используют один и тот же шаблон HTML, так что их структуры очень похожи или одинаковы. Цель – поиск совпадений текущей и предыдущих страниц (это будут области со вспомогательной информацией, такой как навигация, например), несовпадающие элементы и есть записи с существенной информацией.

Глава 1. Описание инструментария

В этой главе будут описаны основные алгоритмы работы приложения.

1.1. Извлечение данных

В данной работе для извлечения информации с веб-страниц применен разработанный метод DSE+, основанный на алгоритме DSE. Применение метода обосновано тем, что в качестве источника используются не произвольные веб-страницы, а страница электронной библиотеки, где все они имеют одинаковую структуру. Ниже описан классический алгоритм DSE.

Работа алгоритма DSE:

1. Выбор страниц для извлечения на основе анализа URL-ссылок. Для определения образцов страниц вводится функция URL подобия US (URL similarity):

$$US(i,\ j) = \ \alpha * \frac{pre(i,\ j)}{len(i,\ j)} + \beta * \frac{pre(i,\ j)}{len(j)} + \gamma,$$

$$\gamma = \begin{cases} 1, \text{если } DN_i = DN_j \\ \dfrac{occurrence\ of(DN_j)}{link\ occurrence\ of(P_i)} \end{cases},$$

где *pre(i, j)* – количество общих символов у url_i и url_j; *len(i)* – длина url_i; нормирующие множители $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$, $\alpha + \beta = 1$ (для нормализации *US(i, j)* $\alpha = \beta = 0{,}5$); DN_j – домен url_j, P_i – страница url_i; *occurrence of* (DN_j) – количество появлений домена на странице. Если *US(i, j)* высока, тогда страница url_i и страница url_j расположены рядом друг с другом, и, возможно, url_j будет хорошим образцом страницы url_i.

2. Выбранные на первом этапе две страницы преобразуются в деревья DOM (Document Object Model), т.е. представляются в виде HTML-кода.

3. Алгоритм ищет совпадающие узлы, согласование которых основано на рекурсивной функции подобия, учитывающей тег, атрибуты и дочерние элементы каждого узла.

   Основная идея состоит в том, чтобы пересечь деревья используя порядок глубин и сравнить их по узлу от корня до листьев, совпадающие узлы удалить. Несколько функций, используемых при сравнении узлов:

   - *RightSibling (parentnode*, $node_i$*)* возвращает правого брата $node_i$, который разделяет *parentnode* с $node_i$.
   - *TagName* ($node_i$) возвращает имя тега $node_i$.
   - Attr ($node_i$) переводит атрибут $node_i$ если это <A> или <IMG>, в противном случае он возвращает null.
   - *Same*($node_i$, $node_j$) = 1, если *TagName* ($node_i$) = *TagName* ($node_j$) и *Attr* ($node_i$) = *Attr* ($node_j$), иначе = 0.

Ниже приведен псевдокод алгоритма согласования деревьев:

```
Matching algorithm: Match (Root_i, Root_j)
  If Same (Root_i, Root_j) = 0, then return 0;
  Else
    If Root_i имеет потомка and Root_j имеет потомка and они имеют
    одинаковое количество потомков
    Then
      For w:=1, ..., m do
        For v:=1,..., n do
          If (Match (child (Root_i, w), child (Root_j, v)) = 1)
          Then
            RemoveChildren (Root_i, w);
            RemoveChildren (Root_j, v);
            Break;
      If все потомки удалены
      Then return 1;
      Else return 0;
    Else
      If Root_i не имеет потомков and Root_j не имеет потомков
      Then return 1;
      Else return 0;
```

Алгоритм работает рекурсивно. Если два узла являются внутренними и совпадают, тогда алгоритм будет опускаться на один уровень, чтобы сопоставить их потомков (от левого к правому). Если узлы являются листьями и совпадают, то они будут удалены из деревьев, иначе алгоритм возвращается

к родительскому узлу и продолжает сравнивать других потомков, если таковые имеются. Два узла-родителя будут удалены, если все их потомки были удалены.

Для целей текущего исследования была разработана модификация алгоритма DSE, названная DSE+. Далее рассматриваются изменения относительно классической версии и обоснования их введения.

В силу того, что анализируемые источники не произвольные веб-страницы, а страницы электронных библиотек, и темы страниц четко заданы пользователем (см. главу 3), возможно использование структуры поискового запроса. Данная структура имеет вид: https://name/search?q=key+word, где name – название сайта, key word – слова поиска. Таким образом, выполнение первого шага алгоритма DSE не является необходимым, время отработки как алгоритма, так и всей программы в целом значительно уменьшается.

В качестве базового средства формирования метаданных для описания широкого класса цифровых объектов обычно упоминается Дублинское ядро метаданных (Dublin Core Metadata Element Set) [8]. Основные элементы, включенные в формат Dublin Core это: Title, Creator (Personal name, CorporateName, Address), Subject, Description (Abstract), Date, Type, Source, Language. Именно поэтому возможно дополнение к методу DSE информации о ключевых тегах – каждое поле записи будет содержать все перечисленные сведения.

Поскольку все записи о публикациях на странице поиска в электронной библиотеке являются потомками одного и того же узла, нет необходимости сравнивать их теги на двух страницах. Значит, можно проводить сравнение узлов в html до тех пор, пока алгоритм не дойдет до ключевых тегов (названия статьи – Title, авторов – Author и тд.). В результате глубина рекурсии алгоритма DSE сократиться, а следовательно, сократится и время работы.

Таким образом, модифицированный алгоритм DSE+ можно записать в

виде:

1. Получение доступа к страницам поиска по ключевым словам, введенным пользователем.
2. Преобразование двух полученных страниц в деревья DOM (Document Object Model), т.е. представление в виде HTML-кода.
3. Поиск совпадающих узлов, согласование которых основано на рекурсивной функции подобия, учитывающей тег, атрибуты и дочерние элементы каждого узла.

   Основная идея состоит в том, чтобы пересечь деревья используя порядок глубин и сравнить их по узлу от корня до листьев, совпадающие узлы удалить.

   Поиск производится до тех пор, пока анализируемые узлы не являются тегами статей (paper/title и т.д.)
4. Сохранение текстовой информации. После шага 3 две страницы являются «полностью отчищенными» от навигационной и рекламной информации и содержат только записи о статьях. На 4 шаге из html извлекается вся текстовая информация и записывается в массив для каждой статьи.

Итогом работы алгоритма является текстовый массив, содержащий в каждом своем элементе всю информацию о конкретной статье.

1.2. Предобработка данных

После первого этапа – извлечения данных – имеется текстовый массив, в каждой строке которого содержится вся информация о статьях: название, список авторов, ссылка на полный текст. Для дальнейшего анализа (см. пункт 1.3) массив необходимо обработать, представив в более удобном виде.

Данные записываются в виде двух массивов, в первом будут сведения о каждом из авторов: его фамилия имя, компания, количество публикаций и

ссылки на публикации (Рис. 1), во втором – взаимосвязи авторов друг с другом: фамилия и имя авторов, их компании и статьи, связывающие их (Рис. 2).

Автор	Компания	Количество статей	Ссылки
Tewari, R.	Petronas	2	link1, link2
Chen, Z.	University of Calgary	3	link1, link2, …
Lemouzy, P.	Beicip	2	link1, link2
Rotondi, M.	Eni SpA	3	link1, link2, …
Sorin, D.	Rhodia	2	link1, link2
Al-Mutairi, S.	Saudi Aramco	2	link1, link2
Lefebvre, C.	IFP	2	link1, link2
Puntervold, T.	University of Stavanger	2	link1, link2
Bourrel, M.	Total E&P	2	link1, link2
Nakashima, T.	Abu Dhabi Company	2	link1, link2

Рисунок 1. Таблица авторов

Порой, при публикации статьей, авторы указывают различные по написанию, но эквивалентные названия компаний, в которых они работают. Во избежание дублирования информации необходимо унифицировать написание компаний, для этого был разработан словарь на основе выборки из 5000 статей на различные темы в нефтедобывающей отрасли. В словарь вошли все перечисленные компании, в которых работали авторы публикаций, и указаны их эквивалентные названия. При обработке данных, извлеченных с онлайн ресурса, все равнозначные названия заменяются на одно и то же обозначение – определение из словаря.

1.3. Анализ данных

В связи с тем, что цель анализа – выделение центров компетенций, необходимо отфильтровать записи. Для этого вводится мера оценки авторов по количеству публикаций *N*; поскольку автор, имеющий мало публикаций в данной области исследований, скорее всего, менее компетентен в ней, необходимо исключить записи о подобных авторах.

Автор, компания	Соавтор, компания	Общие статьи
Al-Yafei, A.,ADNOC	Levitt, D.,Total E&P	EOR Screening for Ekofisk
Moreno, J.,Schlumberger	Moreno, J.,Schlumberger	EOR Screening for Ekofisk
Al-Mayyan, H., Kuwait Oil Company	Al-Mayyan, H., Kuwait Oil Company	EOR Screening for Ekofisk
Bin Ngah, A., WNPOC	Bin Ngah, A., WNPOC	EOR Screening With an Expert System
Bourrel, M.,Total E&P	Bourrel, M.,Total E&P	Methodology for Miscible Gas Injection EOR Screening
Jouenne, S.,Total E&P	Klimenko, A.,Total E&P	Bayesian Network Analysis as a Tool for Efficient EOR Screening
Bourrel, M.,Total E&P	Levitt, D.,Total E&P	Methodology for Miscible Gas Injection EOR Screening
Abdullah, M., Kuwait Oil Company	Pathak, A., Kuwait Oil Company	Comprehensive EOR Screening and Pilot Test for Thar Jath Heavy Oil Field, Sudan
Afra, S.,Texas A&M University	Tarrahi, M.,Shell	Comprehensive EOR Screening and Pilot Test for Thar Jath Heavy Oil Field, Sudan
Al-Murayri, M., Kuwait Oil Company	Abdullah, M., Kuwait Oil Company	Comprehensive EOR Screening and Pilot Test for Thar Jath Heavy Oil Field, Sudan

Рисунок 2. Таблица пар соавторов

Условие допуска данных для дальнейшего анализа выглядит следующим образом:

$$N > \min_i N_i,$$

где N_i – количество статей автора i, $i \in \overline{1, \dots, n}$, $n -$ количество авторов.

Для устранения случаев «включения» в соавторство, вводится мера A (от англ. authorship), которая оценивает отношение количества соавторов и публикаций для каждого автора, так как он не является компетентным, имея, например, одну статью, но с восемью соавторами.

Формула расчета меры «авторства»:

$$A_i = \frac{count(publications_i)}{count(coauthors_i)}.$$

Для отбора компетентных авторов на меру вводится ограничение:

$$A_i > \frac{1}{5},$$

то есть, автор, у которого в среднем на публикацию приходится 5 и более соавторов не допускается для дальнейшего анализа.

Наглядное и информативное представление центров компетенций обеспечивается графом связности авторов, который строится на основе списка вершин и списка ребер, задающего их взаимное расположение в пространстве. Для того, чтобы визуализация была более информативной, необходимо исключить множественные пересечения ребер графа. Это можно достичь двумя способами: преобразованием ребер, например, в кривые B-сплайны [8], или концентрацией вершин по связям, например, минимизация реберных переходов [9] или усреднение всех ребер графа. Первый тип из этих алгоритмов будет влиять на связи вершин, но не на их расположение на рисунке, поскольку изменяет только отображение ребер (Рис. 3).

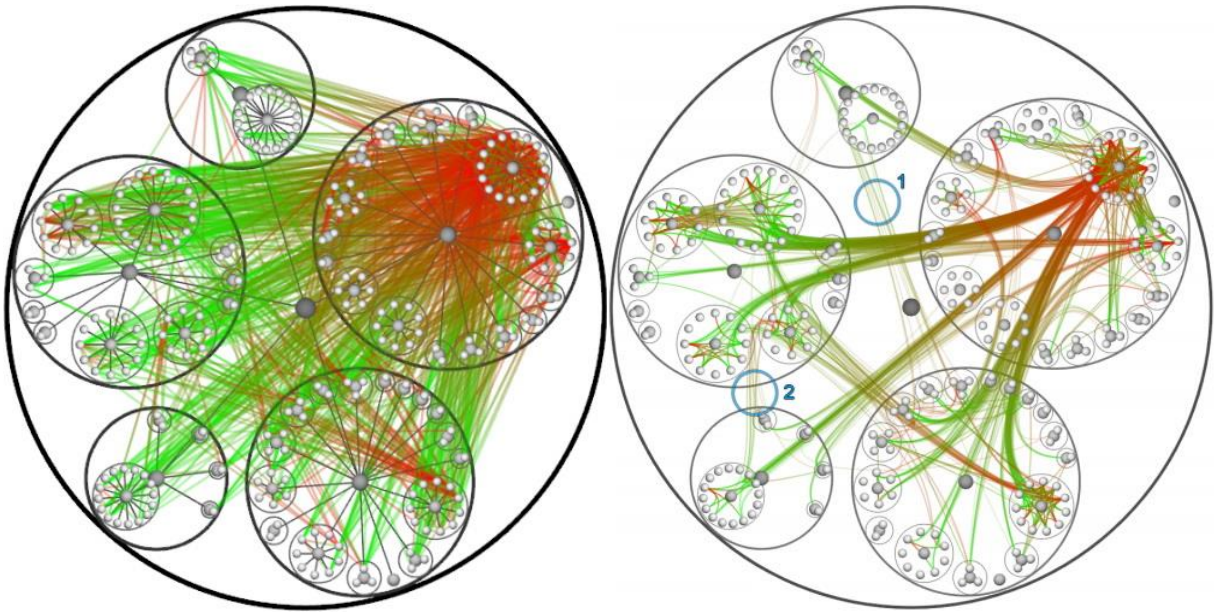

Рисунок 3. Сравнение ребер: прямые и B-сплайны

При использовании второго типа алгоритмов – изменение расположения вершин, возможно локализовать все вершины одного из центров компетенций,

поскольку эти вершины наиболее связаны. Например, один из алгоритмов, основанный на физических аналогиях – силовой (force-directed). Цель состоит в том, чтобы расположить узлы графа в двумерном пространстве так, чтобы все ребра были более или менее одинаковой длины, и как можно меньше пересекались между собой. Это достигается путем назначения сил между множеством ребер и узлов, основанных на их относительных положениях, и использования этих сил либо для имитации движения ребер и узлов, либо для минимизации их энергии. В данной работе применен метод Камада и Каваи [10], результат работы которого описан в главе 3.

Согласно алгоритму Камада-Каваи каждая пара вершин связана пружинкой, идеальная длина которой равна наименьшему расстоянию между вершинами графа. Потенциальная энергия такой системы с *n(n-1)/2* ребром и размещением вершин p есть разница между евклидовым расстоянием и расстоянием в графе:

$$E_{KK} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (|p_i - p_j| - l_{ij})^2 ,$$

где l_{ij} – идеальная длина пружины между i и j, $k_{ij} = K/d_{ij}^2$ – сила пружины, d_{ij} – расстояние между i и j в графе. Алгоритм ищет параметры, минимизирующие функционал E. Итеративный метод находит вершину с наибольшим градиентом и смещает ее, при этом позиции остальных вершин неизменны. Останавливается алгоритм тогда, когда градиент меньше порогового значения [11].

Глава 2. Реализация

В качестве рабочей машины использовался компьютер с процессором Intel Core i5 – 4570 CPU @ 3.20 GHz 8 Гб. Для реализации проекта был выбран язык программирования Python (ver. 3.5) и следующие библиотеки: requests, BeautifulSoup – для извлечения информации с веб-страниц, igraph [12], plotly [13] – для построения графа, flask – для интерфейса.

2.1. Архитектура

Для разработки системы было выбрано веб-приложение по причине удобного доступа с разных платформ и отсутствия необходимости в процессе установки программ, что позволит максимально упростить работу пользователям.

По просьбе заказчика данные (массив статей, списки вершин и ребер) и результаты программы (графы) хранятся на стороне пользователя. Это обеспечивает возможность добавления/исправления данных вручную, быстрый доступ к данным для сторонней работы.

2.2 Логика работы

Работа приложения подразделяется на три уровня: взаимодействие с пользователем, сбор данных, обработка данных.

Общая структура работы:

1. Первый этап: извлечение и предобработка данных (исходный код программы представлен в приложении 1).
2. Второй этап: построение графа (исходный код представлен в приложении 2).

Алгоритм работы в виде блок-схемы представлен на рисунке 4. На рисунке 5 изображен интерфейс приложения. Исходный код веб-шаблона и

взаимодействия с пользователем представлен в приложении 3.

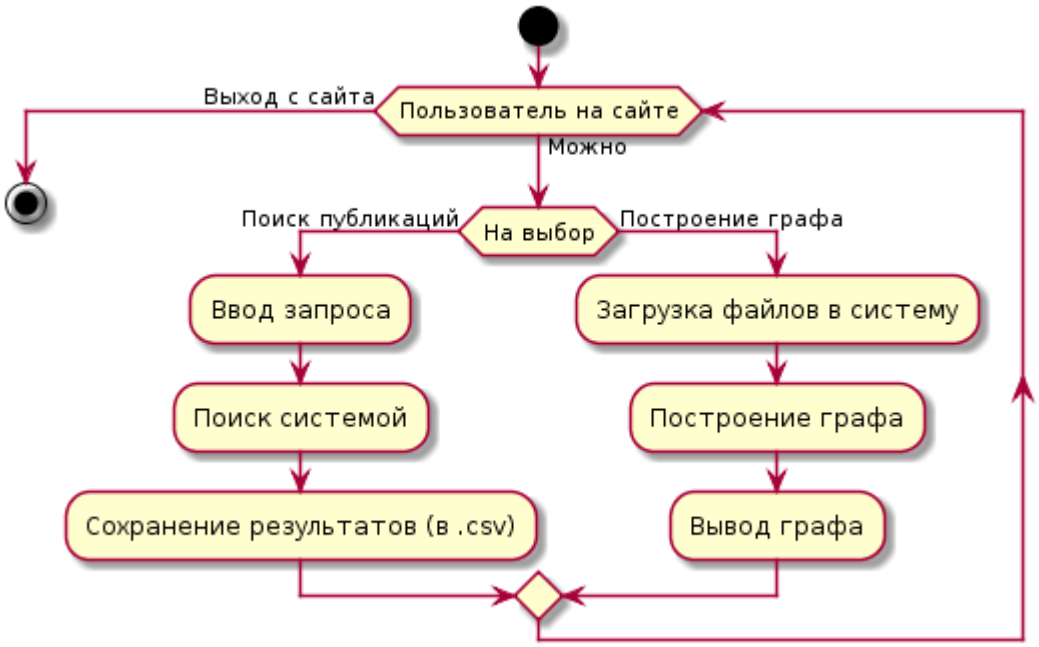

Рисунок 4. Логика работы программы.

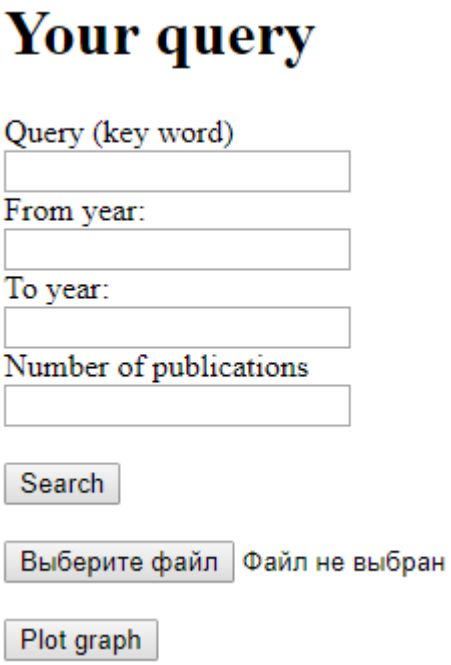

Рисунок 5. Веб-страница приложения.

Глава 3. Обзор работы приложения

Для получения данных о конкретном методе увеличения нефтедобычи пользователь вводит в поисковую форму ключевое слово-характеристику пласта, на котором ведется производство, может ограничить поиск следующими критериями: с какого и по какой год искать публикации, их количество. После заполнения информации, пользователь нажимает кнопку поиска и выполняется извлечение, обработка и запись данных в файлы, которые загружаются на компьютер. Далее есть возможность внести изменения и правки в выгруженные файлы.

На втором этапе происходит построение интерактивного графа связности авторов, отражающий также принадлежность авторов к компании (цвет) и количество статей у авторов – размер вершины. Подписи вершин содержат информацию об авторе, количестве публикаций и компании.

В качестве объекта тестирования была предложена онлайн-библиотека технической литературы для исследования и добычи нефти и газа OnePetro.org [14]. На рисунке 6 представлен итог работы программы для запроса в целом о скрининге МУН (запрос “eor screening”). По данному запросу можно выделить три крупных: компании Schlumberger, Total E&P, Petronas; и два средних центра компетенций: Tiorco и IFP. При этом наибольшее количество публикаций у авторов из компаний: Schlumberger, Petronas, Saudi Aramco, Shell и Missouri University of Science and Technology.

На рисунке 7 представлен итог работы программы для залежи известняка (запрос “lime”). Как видно по рисунку для публикаций по данному запросу в библиотеке OnePetro нельзя выделить четких центров компетенций. При этом есть единственный автор с наибольшим количеством публикаций: Nasr-El-Din H. Из компании Texas A&M University.

На рисунке 8 представлен итог работы программы для запроса о

солености залежи. Данное изображение выявляет несовершенность изображения центров компетенций в виде графов фиксированной размерности – происходит перекрывание вершин-авторов. Данная проблема решается путем приближения интересующей области (Рис. 9).

В ходе тестирования было выявлено, что несмотря на предобработку названий компаний, недостаточно примененного словаря, поэтому в дальнейшем необходимо разработать алгоритм анализа и замены эквивалентных наименований. Также, поскольку тестирование проводилось на одном источнике статей, то, при применении фильтра авторов, количество записей о статьях сокращается и не совсем объективным становится анализ на выявление центров компетенций. Данная проблема будет решена при использовании двух и более источников извлечения информации.

В целом программа показывает хорошие результаты: среднее время работы 10 сек, количество обрабатываемых записей 1500.

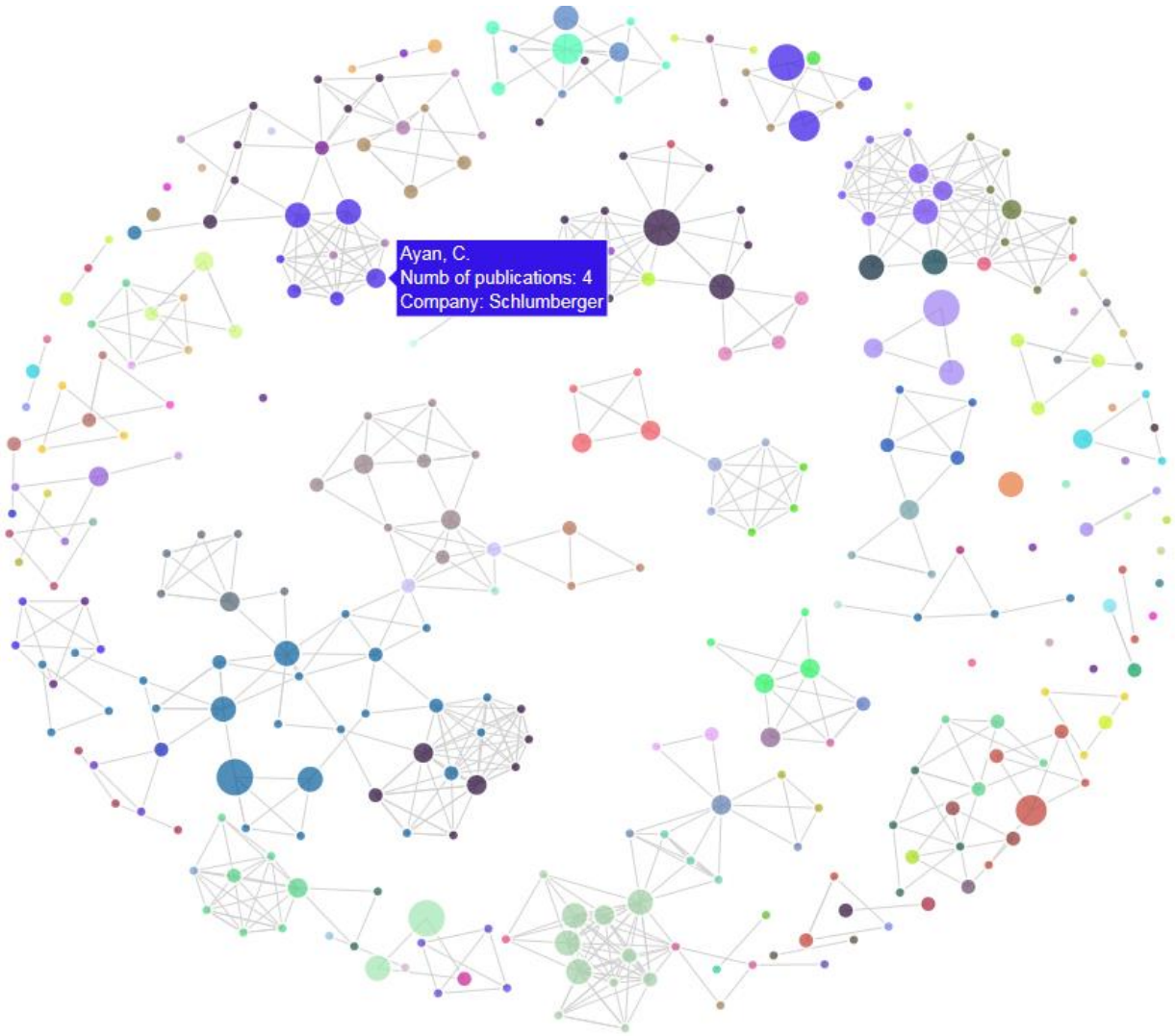

Рисунок 6. Граф связей авторов для запроса "eor screening"

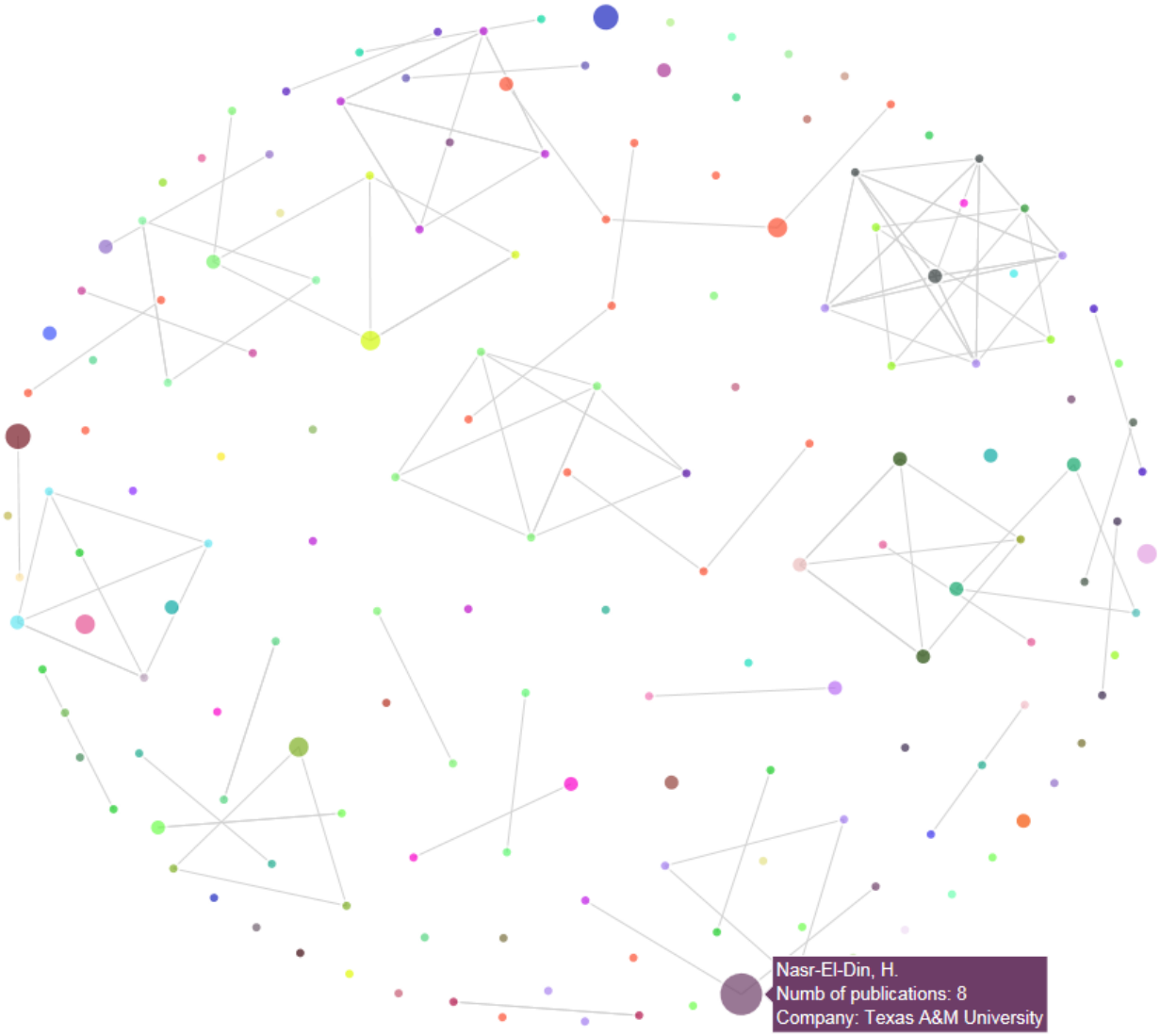

Рисунок 7. Граф связей авторов по теме “lime”

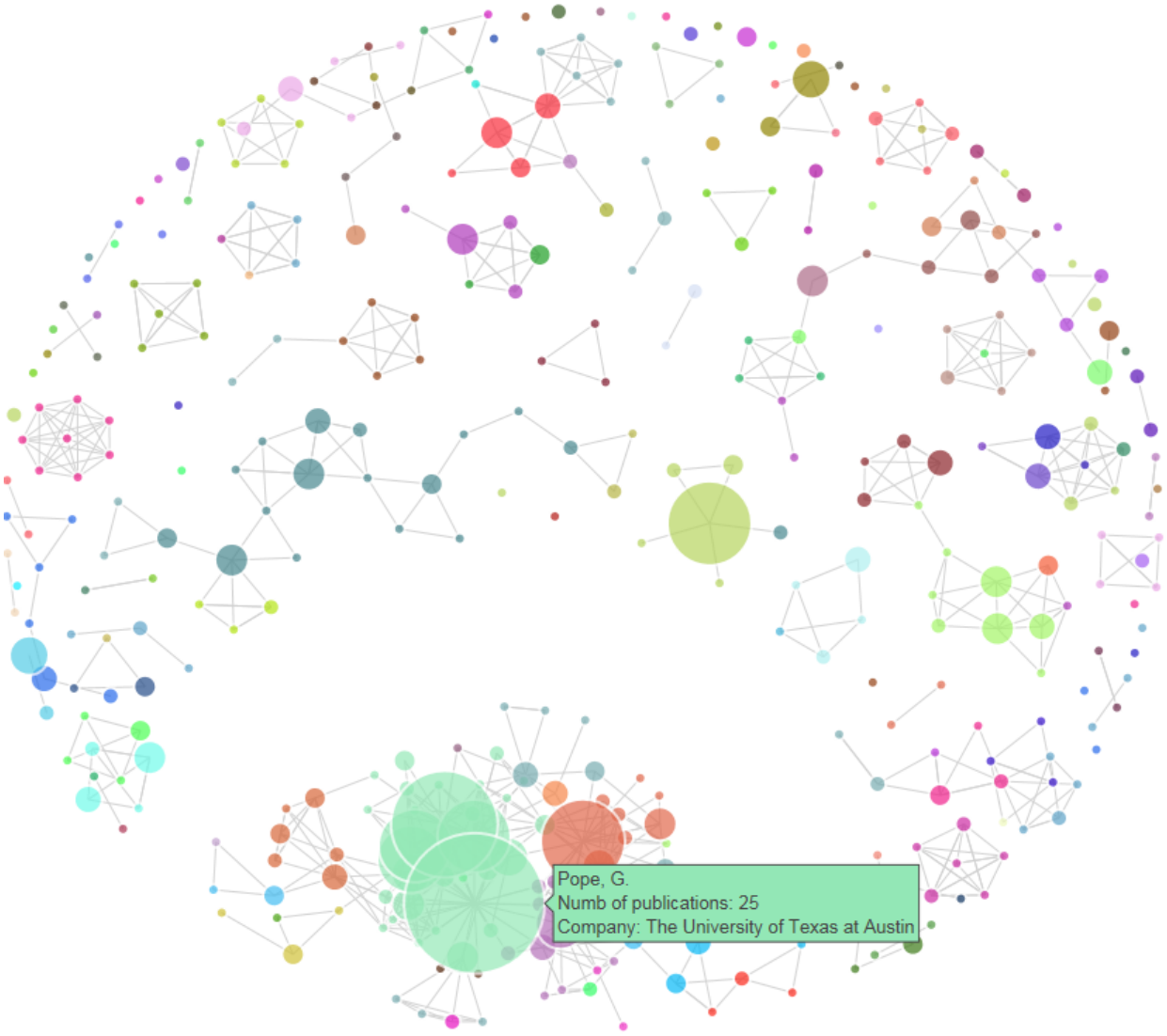

Рисунок 8. Граф соавторов для соляных месторождений

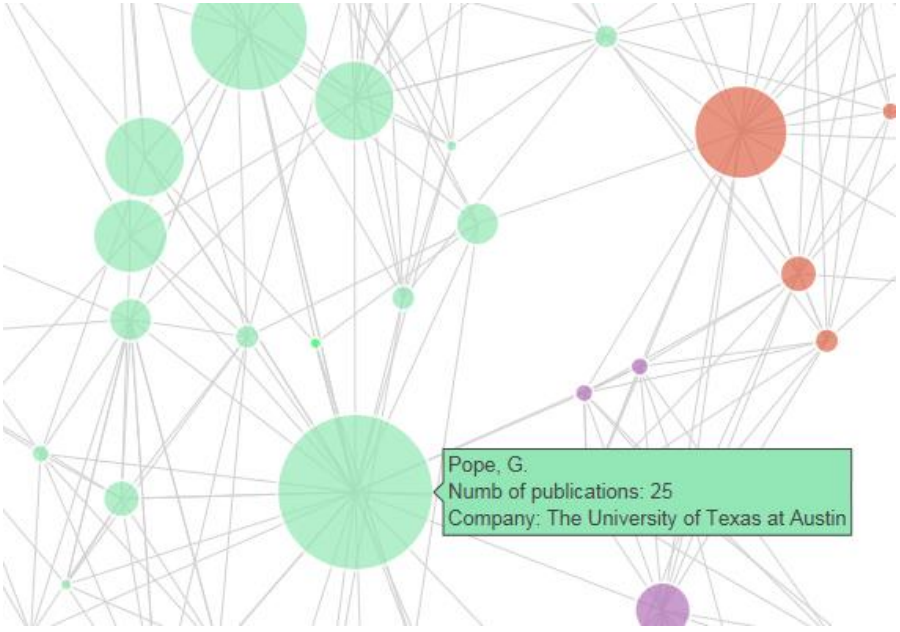

Рисунок 9. Приближение области

Выводы

В рамках данной работы была поставлена цель – повышение эффективности скрининга технологий МУН. Для достижения данной цели были поставлены задачи извлечения данных из онлайн библиотеки и разработки программного обеспечения, автоматизирующего использование обозначенных данных.

С учетом существующих решений, посредством самостоятельного изучения предметной области был разработана модификация алгоритма извлечения информации с веб-страниц DSE+ и предложена возможность их анализа на применимость методов повышения нефтедобычи. Было разработано веб-приложение для пользователей с возможностью самостоятельного дополнения и исправления данных, а также их визуализации.

Одно из возможных развитий приложения – извлечение данных из веб-ресурсов и их предобработка для последующей работы со сторонними приложениями. Другое возможное развитие, помимо сферы скрининга, – поиск центров компетенций по запрашиваемой тематике независимо от области знаний.

Заключение

С целью автоматизации подготовки данных для скрининга методов повышения нефтедобычи была разработана система извлечение и анализа данных из электронных ресурсов.

Разработанная система может быть удобным инструментом для повышения качества и скорости работы по выбору технологий МУН. Приложение значительно снизит время на обработку больших объемов информации сотрудниками нефтедобывающей компании, что позволит сфокусироваться на более приоритетной части скрининга – выборе технологии. Кроме того, хранение данных в электронном виде позволит значительно снизить затрачиваемые усилия на повторный поиск.

Список литературы

1. Кузьмичев Н. П. КЭС – новый подход к повышению рентабельности добычи нефти // Бурение и нефть, 2005. № 6. С. 16-17.
2. Тяжелые нефти в России [Электронный ресурс]: URL: http://one_vision.jofo.ru/241887.html (дата обращения 19.12.2017).
3. Линч К. Четвертая парадигма Джима Грея и формирование архива науки // Четвертая парадигма / под ред. Хея Т., Тэнсли С., Толле К. Microsoft research. 2014. С. 175–182.
4. Sleiman H. A., Corchuelo R. A Survey on region extractors from web documents // IEEE Transactions on Knowledge and Data Engineering. 2013. Vol. 25. No 9. P. 1960–1980.
5. Embley D.W., Jiang Y.S., Ngy Y.-K. Record-Boundary Discovery in Web Documents // ACM SIGMOD Record. 1999. Vol. 28. No 2. P. 467–478.
6. Buttler D., Liu L., Pu C. A fully automated object extraction system for the World Wide Web // Proc. Int'l Conf. Distributed Computing Systems (ICDCS). 2001. P. 361–370.
7. Wang J., Lochovsky F.H. Data-rich section extraction from HTML pages // Proc. Third Int'l Conf. Web Information Systems Eng. (WISE). 2002. P. 313–322.
8. Holten D. Hierarchical edge bundles: visualization of adjacency relations in hierarchical data // IEEE Transactions on Visualization and Computer Graphics. 2006. Vol. 12. No 5. P. 741–748.
9. Huang W., Hong S., Eades P. Layout Effects on Sociogram Perception // Proc. 12th Int'l Symp. Graph Drawing. 2006. P. 262–273.
10. Kamada T., Kawai S. An algorithm for drawing general undirected graphs // Information Processing Letters. 1989. Vol. 31. No 1. P. 7–15.
11. Апанович З.В. Современные силовые алгоритмы для визуализации информации большого объема // Институт систем информатики им. А.П.

Ершова СО РАН. С. 164–171.

12. Igraph [Электронный ресурс]: URL: http://igraph.org/python/ (Дата обращения 12.04.2018).
13. Plotly [Электронный ресурс]: URL: https://plot.ly/ (Дата обращения 12.04.2018).
14. OnePetro.org [Электронный ресурс]: URL: https://www.onepetro.org/ (дата обращения 20.03.2018).
15. Song Y., Wei R. Research on application of data mining based on FP-growth algorithm for digital library // Second International Conference on Mechanic Automation and Control Engineering (MACE). 2011. P. 1525–1528.
16. Wu J., Williams K., Chen H., Khabsa M., Caragea C., Ororbia A., Jordan D., Lee Giles C. CiteSeerX: AI in a Digital Library Search Engine // Twenty-Sixth Annual Conference on Innovative Applications of Artificial Intelligence. 2014. P. 2930–2937.
17. Peng F., McCallum A. Accurate Information Extraction from Research Papers using Conditional Random Fields. 2004.
18. Wu J., Williams K., Choudhury S. R., Khabsa M., Lee Giles C. CiteSeerX: Scholarly Big Data Information Extraction and Integration in the CiteSeer Digital Library // IEEE 30th International Conference on Data Engineering Workshops (ICDEW). 2014. P. 68–73.
19. Herlocker J., Jung S., Webster J. Collaborative Filtering for Digital Libraries. 2003.

Приложения

Приложение 1

Ниже представлен исходный код этапов извлечения и предобработки данных на примере сайта onepetro.org.

```
def pre_processing(query, f_year, t_year, numb):
    query = query.replace(" ", "+") # заменяем пробел знаком +
    param =
query+'&peer_reviewed=&published_between=&from_year='+f_year+'&to_year='+t_ye
ar+'&rows=25&start='  # формируем параметры запроса
    url = 'https://www.onepetro.org/search?q='+param+str(0) # доступ к
странице запроса
    html = requests.get(url) # извлечение html кода
    N= re.findall(r'<h2>[a-zA-Z:\s.]+([\s,0-9]+)\s', html.text)[0] #
извлечение строки, гда содержится информация о числе результатов
    N = N.replace(",", "") # удаление запятых
    N = N.replace(" ", "") # удаление пробелов
    # если введено ограничение на количество статей, то учитываем его
    if numb != '':
        if int(numb)<int(N):
            N = numb

    auths = []
    forgraph = []
    comp = []
    # для каждой страницы (по 100 статей) выполняем
    for ii in range(0, math.ceil(int(N)/100)):
        url = 'https://www.onepetro.org/search?q='+param+str(ii*100) # доступ
к старнице
        html = requests.get(url) # извлечение html
        soup = BeautifulSoup(html.text, 'html.parser') #извлекаем текст из
html
#         soup = BeautifulSoup(html, 'html.parser') # для тестирования на
текстовом файле
        papers = soup('div', {"class":"result-item"}) # извлекаем все статьи
на странице
        # для каждой публикации (они все ограничены тегом "result-item")
        for p in papers:
            authors=[] # массив для записи всех авторов статьи

            #extract authors
            auth = p('div', {"class":"result-item-author"}) #для каждого
класса с пометкой "result-item-author" автора
            for tag in auth:
                tag = re.sub(r'\s+',' ',tag.text) # удаляем все лишние
пробелы и переносы
                tag = tag.replace('&amp;', '') #удаляем служебные символы &,
числа и тд
                tag = tag.replace('&amp;apos;', '')
                tag = tag.replace('&apos;', '')
                tag = tag.replace('Ø', 'O')
                tag = tag.replace('ª', 'a')
```

```
                tag = tag.replace('©', 'c')
                tag = re.sub(r'#[0-9]*;', '', tag)
                # если есть подходящие под шаблон строки
                if (re.findall(r"^ ([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, \S{1})[À-ÿa-zA-Z-_;()'&.\s]*,", tag) != []):
                    authors.append(re.findall(r'^ ([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-
ÿa-zA-Z0-9-.,&\s\S]*)', tag)[0])
                    company = ''.join(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-
ÿA-Za-z-_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]+)",
tag))
                    company = rename(company)
                    auths.append(re.findall(r"^ ([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-
Za-z-_;()'&.\s]+, \S{1})[À-ÿa-zA-Z-_;()'&.\s]*,", tag)[0]+'.'+','+company)
            # составление комбинаций авторов (апример для А,Б,В получим АА,
АБ, АВ, ББ, БВ, ВВ)
            A = itertools.combinations_with_replacement(authors,2)
            for i in A:
                l=[]
                # компания и авор первого элемена из комбинации
                company = ''.join(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-
z-_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]+)", i[0]))
                company = rename(company)
                l.append(re.findall(r"^([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, \S{1})[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿa-zA-Z-_;()'&.\s]*,",
i[0])[0]+'.'+','+company)
                # компания и авор второго элемена из комбинации
                company = ''.join(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-
z-_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]+)", i[1]))
                company = rename(company)
                l.append(re.findall(r"^([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, \S{1})[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿa-zA-Z-_;()'&.\s]*,",
i[1])[0]+'.'+','+company)
                # запись в общий массив
                forgraph.append(l)
    auths = OrderedDict(sorted(dict(Counter(auths)).items(),
key=itemgetter(1))) # словарь со всеми авторами - без повторов, ключь -
фамилия, значение - количество статей
    reb = auths[(list(auths)[len(auths)*2//10])] # порог, по которому
отсекаем авторов (80%)
    new_auths = {k: v for k, v in auths.items() if v >reb} # новый список
авторов - без "малостатейных"

    pairs = set() # перемнная для записи пар автор-соавтор, которые вошли в
80%
    for i in forgraph:
        # условие что и автор и соавтор входят в 80%
        if ((i[0] in list(new_auths)) & (i[1] in list(new_auths))):
            pairs.add(tuple(i))
            # запись в перемнную компаний
            comp.append(''.join(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]+)", i[0])))
            comp.append(''.join(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]+)", i[1])))

    comp = dict(Counter(comp)) # словарь для компаний
    color = [] # переменная для записи цветов
```

```
    # цикл для задания рандомных цветов (нужно бы что-то другое найти, чтоб
они не были блеклыми)
    for c, v in comp.items():
        r = lambda: random.randint(0,255)
        r = '#%02X%02X%02X' % (r(),r(),r())
        color.append(r)

    param = query
    if f_year != '':
        param = param+'_'+f_year
    if t_year != '':
        param = param+'_'+t_year
    if numb != '':
        param = param+'_'+numb
    # запись в csv файл вершин
    out = 'node.csv'
    outfile = open(out,'w', newline='')
    writer=csv.writer(outfile, delimiter =';')
    writer.writerow(['Автор', 'Компания', 'Количество статей', 'Ссылки'])
    for k, v in new_auths.items():
        n = list(comp).index(re.findall(r"^[àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-_;()'&.\s]+,([\s\S]*)", k)[0])
        writer.writerow([re.findall(r"^([àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿA-Za-z-
_;()'&.\s]+, [àèéìíîòóùúÀÈÉÌÍÎÒÓÙÚÀ-ÿa-zA-Z-_;()'&.\s]+),",k)[0],
re.findall(r"^[A-Za-z-_;()'&.\s]+, [A-Za-z-_;()'&.\s]+,([\s\S]*)", k)[0],
str(v),  str(color[n])])
    outfile.close()

    # запись в csv файл ребер
    out = 'edges.csv'
    outfile = open(out,'w', newline='')
    writer=csv.writer(outfile, delimiter =';')
    writer.writerow(['Автор, компания', 'Соавтор, компания', 'Общие статьи'])
    for f in pairs:
        writer.writerow(f)
    outfile.close()
```

Приложение 2

Ниже представлен код этапа анализа полученных данных.

```
def plot_graph(param):
    print('p ', param)
    df_nodes = pd.read_csv('node.csv', encoding='mac_roman', delimiter =';')
# чтание из файла с вершинами
    reader = csv.DictReader(open("edges.csv"), dialect="excel",
delimiter=';') # чтение из файла со связями
    graph = Graph.DictList(vertices=None, edge_foreign_keys=('Author,
company', 'Co-author, company'), edges=reader) # составление графа
    nodes=list(graph.vs['name']) # вершины
    edges = [e.tuple for e in graph.es] # ребра
    layt=graph.layout('kk') # задание координат вершин в зависимости от из
связности по алгоритму Kamada-Kawai
    N=len(nodes) # число вершин
    Xn=[layt[k][0] for k in range(N)] # задание координат для вершин по оси x
    Yn=[layt[k][1] for k in range(N)] # задание координат вершин по оси y
```

```
    Xe=[] # вектор-переменная для записи y-координат отрезков-связей
    Ye=[] # вектор-переменная для записи x-координат отрезков-связей
    for e in edges:
        Xe+=[layt[e[0]][0],layt[e[1]][0], None]
        Ye+=[layt[e[0]][1],layt[e[1]][1], None]

    # создание списка для связей (рёбер)
    edge_trace=Scatter(x=Xe,
               y=Ye,
               mode='lines',
               line=Line(color='rgb(210,210,210)', width=1),
               hoverinfo='none'
               )
    # создание списка для вершин

    node_trace = Scatter(
                    x=[],
                    y=[],
                    text=[],
                    name = [],
                    mode='markers',
                    hoverinfo='text',
                    marker=Marker(
                        color=[],
                        size= [],
                        line=dict(width=2)))
    # заполнение списка для вершин
    # для каждой вершины из файла
    for i in range(len(df_nodes)):
        i_node =
nodes.index(df_nodes['Author'][i]+','+df_nodes['Company'][i]) # определение
индекса вершины в списке node из графа graph
        x, y = Xn[i_node], Yn[i_node] # определение по индексу
соответствующих оординат
        node_trace['x'].append(x) # добавление координаты x
        node_trace['y'].append(y) # добавление координаты y
        node_trace['marker']['size'].append(4*df_nodes['Number of
publications'][i]) # добавление размера пропорционально количеству статей у
автора
        node_trace['marker']['color'].append(df_nodes['Color'][i]) #
добавление цвета в соответствии с компанией
         # извлечение автора для создания подписи
        auth = df_nodes['Author'][i]
        t = auth +'<br>'+'Numb of publications: '+str(df_nodes['Number of
publications'][i])+'<br>'+'Company: '+str(df_nodes['Company'][i])
         # доавление подписи
        node_trace['text'].append(t)
        node_trace['name'].append(df_nodes['Company'][i])

    # координатные оси и их параметры
    axis=dict(showline=False, # hide axis line, grid, ticklabels and  title
          zeroline=False,
          showgrid=False,
          showticklabels=False,
          title=''
          )
    # задание размера отображения
```

```
    width=1000
    height=1000

    # задание расположения графа и подписи
    layout=Layout(title= "Co-authorship network"+\
              "<br> Data source: <a href='https://onepetro.org'>
OnePetro</a>",
        font= Font(size=12),
        showlegend=False,
        autosize=False,
        width=width,
        height=height,
        xaxis=XAxis(axis),
        yaxis=YAxis(axis),
        margin=Margin(
                      l=40,
                      r=40,
                      b=85,
                      t=100,
                      ),
        hovermode='closest',
        annotations=Annotations([
           Annotation(
           showarrow=False,
            text='This igraph.Graph has the Kamada-Kawai layout',
            xref='paper',
            yref='paper',
            x=0,
            y=-0.1,
            xanchor='left',
            yanchor='bottom',
            font=Font(
            size=14
            )
            )
        ]),
    )
    # запись в переменную вершин и ребер
    data=Data([edge_trace, node_trace])
    fig=Figure(data=data, layout=layout)
    # отображение и сохранение в html
    plotly.offline.plot(fig, filename='Coautorship-network-igraph.html')
```

Приложение 3

Ниже представлен код веб-части приложения.

```
param = ''
global param
@app.route("/", methods=['GET', 'POST'])
def search():
    result = '' # переменная для записи результата (поясняющая надпись)
    error = '' # переменная для записи ошибки
    # получаем данные с веб-страницы
    if request.method=='POST':
        # если нажата кнопка поиска
```

```
        query = request.form['query'] # запрос - ключевые слова
        f_year = request.form['f_year'] # год, с которого начать просмотр
        t_year = request.form['t_year'] # год, младше которого не
рассматривать
        numb = request.form['numb']  # количество анализируемых публикаций
        param = query
        if f_year != '':
            param = param+'_'+f_year
        if t_year != '':
            param = param+'_'+t_year
        if numb != '':
            param = param+'_'+numb
        print(param)

        if request.form['submit'] == 'Search':
            # если запрос введен
            if query:
                try:
                    pre_processing(query, f_year, t_year, numb) # вызываем
функцию обработки данных и записи их в файл
                    result = 'csv files are ready' # посяняюзщая надпись
                except ValueError:
                    # ошибка
                    error = 'Please write your query.'
        # если нажата кнопка построения графа
        elif request.form['submit'] == 'Plot graph':
            try:
                plot_graph(param) # вызов функции построения графа
                result = 'Graph is plot' # поясняюща надпись
            except ValueError:
                error = 'Some error with data' # ошибка с данными
     # вывод результатов на веб страницу
    return render_template('search_ui2.html', result=result, error=error)

if __name__ == "__main__":
    app.run(threaded=True)
```

Веб-представление написано на языке html и представлено ниже.

```
<!doctype html>
<title>Graph of authors</title>
<h1>Your query</h1>
<form method="POST">
    <label name='l_query'>Query (key word)</label>
    <br>
    <input type="text" name="query" value="">
    <br>
    <label name='l_f_year'>From year:</label>
    <br>
    <input type="text" name="f_year" value="">
    <br>
    <label name='l_t_year'>To year:</label>
    <br>
    <input type="text" name="t_year" value="">
    <br>
    <label name='l_numb'>Number of publications</label>
```

```
    <br>
    <input type="text" name="numb" value="">
    <br>
    <br>
    <input type="submit" name="submit" value="Search">
    <br>
<br>
    <input type = "file" name = "file" />
<br>
<br>
<input type="submit" name="submit" value="Plot graph">
    {% if result %}
    <p>
        <label name='result'>{{ result }}</label>
    </p>
    {% endif %}
    {% if error %}
    <p>
    <label name="error">{{ error }}</label>
    </p>
    {% endif %}
</form>
```